MAGMA MIC
Optimizing Linear Algebra for Intel Xeon Phi

Stan Tomov
w/ H. Anzt, J. Dongarra, M. Gates, A. Haidar, K. Kabir, P. Luszczek, and I. Yamazaki
Innovative Computing Laboratory
Department of Computer Science
University of Tennessee, Knoxville

ISC'15, Frankfurt, Germany
Intel Booth Presentation
July 13, 2015

THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

IPCC at ICL

LAPACK and
ScaLAPACK

- Standard dense linear algebra (DLA) libraries
- Many applications rely on DLA
- Designed in 80/90's for cache-based architectures

Must be redesigned for modern heterogeneous systems with multi/many-core CPUs and coprocessors.

IPCC at ICL

- **Develop**
  - Next generation LAPACK / ScaLAPACK
  - Programming models, and
  - Technologies

  for heterogeneous

  Intel Xeon Phi-based platforms

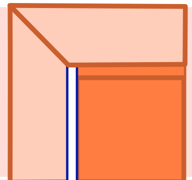
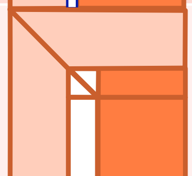
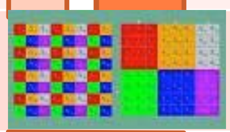
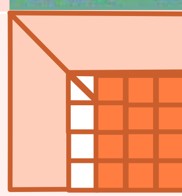
- **Disseminate developments through the MAGMA MIC library**

- **High value proposition**

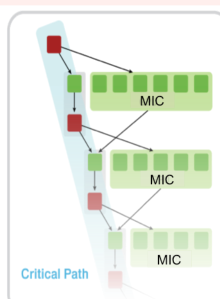
  MAGMA MIC enables ease of use and adoption of Intel Xeon Phi architectures in applications as linear algebra is fundamental to scientific computing

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

A New Generation of Dense Linear Algebra Libraries

Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels

MAGMA
Hybrid Algorithms
(heterogeneity friendly)



Rely on
- hybrid scheduler
- hybrid kernels

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

MAGMA MIC
LAPACK for heterogeneous systems

- **MAGMA MIC**
  - Project on the development of a new generation of HP Linear Algebra Libraries
  - To provide LAPACK/ScaLAPACK on heterogeneous Intel Xeon Phi-based systems
  - Well established project with product disseminated through the MAGMA MIC libraries:

MAGMA MIC 0.3 (2012-11-13)

MAGMA MIC 1.0 (2013-05-03)

MAGMA MIC 1.1 (2014-01-07)

MAGMA MIC 1.2 (2014-09-17)

MAGMA MIC 1.3 (2014-11-15)

MAGMA MIC 1.4 (2015-07-12)

- For heterogeneous, shared memory systems
- Included are the main factorizations, linear system and eigen-problem solvers
- Open Source Software (http://icl.cs.utk.edu/magma)

- **Collaborators**
  - Intel MKL Team
  - UC Berkeley, UC Denver, INRIA (France), KAUST (Saudi Arabia)
  - Community effort, similar to LAPACK/ScaLAPACK

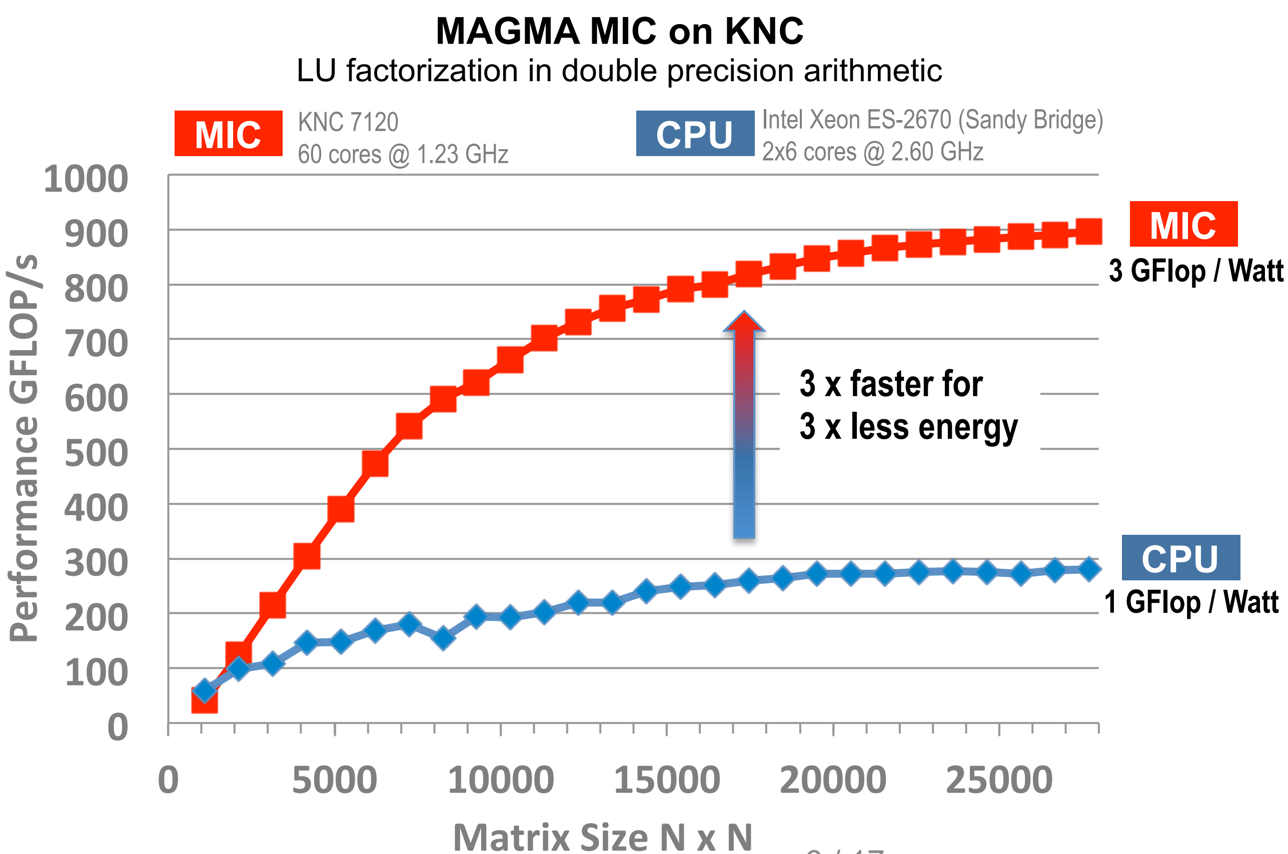
ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Key Features of MAGMA MIC

HYBRID ALGORITHMS

MAGMA MIC uses hybrid algorithms where the computation is split into tasks of varying granularity and their execution scheduled over the hardware components. Scheduling can be static or dynamic. In either case, small non-parallelizable tasks, often on the critical path, are scheduled on the CPU, and larger more parallelizable ones, often Level 3 BLAS, are scheduled on the MICs.

PERFORMANCE & ENERGY EFFICIENCY



FEATURES AND SUPPORT

MAGMA MIC 1.4

- Linear system solvers
- Eigen-problem solvers
- SVD
- CPU/AO interface
- MIC/Native interface
- Multiple precision support
- Mixed-precision iter. refinement solvers
- Multicore and multi-MIC support
- Sparse LA
- LAPACK testing
- Linux

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

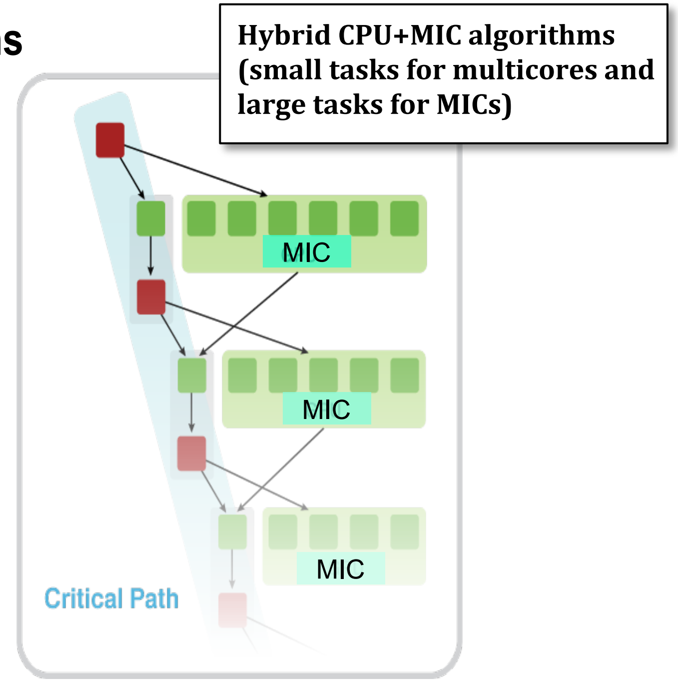
Methodology overview

A methodology to use all available resources:

- MAGMA MIC uses hybrid algorithms
  - Representing linear algebra algorithms as collections of tasks and data dependencies among them
  - Properly scheduling tasks' execution over multicore CPUs and manycore coprocessors

- Successfully applied to fundamental linear algebra algorithms
  - One- and two-sided factorizations and solvers
  - Iterative linear and eigensolvers

- Productivity
  1) High level;
  2) Leveraging prior developments;
  3) Exceeding in performance homogeneous solutions



ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

A Hybrid Algorithm Example

Left-looking hybrid Cholesky

From sequential
LAPACK

```
for( j=0, j<n; j+=nb) {
    jb = min(nb, n-j);
    zherk( MagmaUpper
           jb, j, one, dA(0

    if (j+jb < n)
      zgemm( MagmaCo
             dA(0,j), ldd

     zpotrf( MagmaUpper
     if (info != 0)
       *info += j;

     If (j+jb) < n) {

       ztrsm( MagmaLeft,
              jb, n
     }
}
```

to parallel hybrid
MAGMA

```
for( j=0, j<n; j+=nb) {
    jb = min(nb, n-j);
    magma_zherk( MagmaUpper, MagmaConjTra
                 jb, j, one, dA(0,j), ldda, one, dA
     magma_zgetmatrix_async( jb, jb, dA(j,j), ldd
     if (j+jb < n)
       magma_zgemm( MagmaConjTrans, Magma
                    dA(0,j), ldda, dA(0,j+jb), ldd
     magma_event_sync( event );
     zpotrf( MagmaUpperStr, &jb, work, &jb, info);
     if (info != 0)
        *info += j;
     magma_zsetmatrix_async(jb, jb, work, jb, dA
     If (j+jb) < n) {
       magma_event_sync( event );
       magma_ztrsm( MagmaLeft, MagmaUpper, 
                    jb, n-j-jb, one, dA(j,j), ldda, d
       }
    }
```

MAGMA runtime environment

- Scheduling can be static or dynamic
- Dynamic is based on QUARK
- Uses CUDA streams to offload computation to the GPU

Note:
- MAGMA and LAPACK look similar
- Difference is lines in red, specifying data transfers and dependencies
- Differences can be hidden in a dynamic scheduler making the top level representation of MAGMA MIC algorithms almost identical to LAPACK

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

A Hybrid Algorithm Example

Left-looking hybrid Cholesky

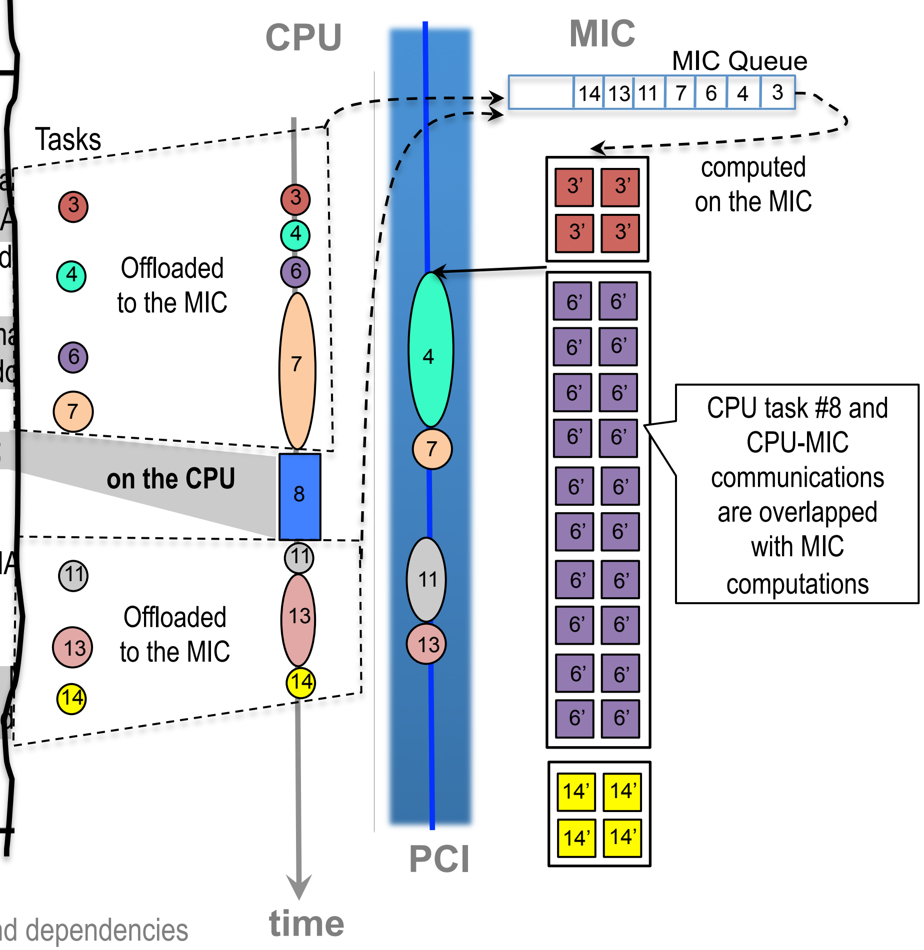
From sequential
LAPACK

```
for( j=0, j<n; j+=nb) {
    jb = min(nb, n-j);
    zherk( MagmaUpper
           jb, j, one, dA(0

    if (j+jb < n)
        zgemm( MagmaCo
               dA(0,j), ldd

    zpotrf( MagmaUpper
    if (info != 0)
        *info += j;

    If (j+jb) < n) {

        ztrsm( MagmaLeft,
               jb, n
    }
}
```

to parallel hybrid
MAGMA

```
1   for( j=0, j<n; j+=nb) {
2       jb = min(nb, n-j);
3       magma_zherk( MagmaUpper, MagmaConjTra
                     jb, j, one, dA(0,j), ldda, one, dA
4       magma_zgetmatrix_async( jb, jb, dA(j,j), ldd
5       if (j+jb < n)
6           magma_zgemm( MagmaConjTrans, Magma
                         dA(0,j), ldda, dA(0,j+jb), ldd
7       magma_event_sync( event );
8       zpotrf( MagmaUpperStr, &jb, work, &jb, info);
9       if (info != 0)
10          *info += j;
11      magma_zsetmatrix_async(jb, jb, work, jb, dA
12      If (j+jb) < n) {
13          magma_event_sync( event );
14          magma_ztrsm( MagmaLeft, MagmaUpper,
                         jb, n-j-jb, one, dA(j,j), ldda,
        }
    }
```

MAGMA MIC runtime environment

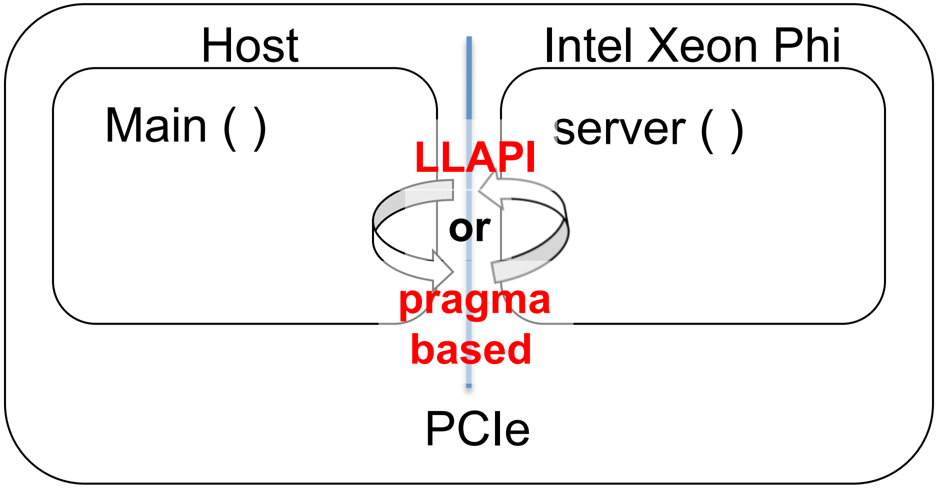


Note:
- MAGMA and LAPACK look similar
- Difference is lines in red, specifying data transfers and dependencies
- Differences can be hidden in a dynamic scheduler making the top level representation of MAGMA MIC algorithms almost identical to LAPACK

ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Programming models

- **We developed two APIs for offloading work to MIC:**



Both APIs have the same interface and abstract low level programming details

LLAPI based
- A server runs on the MIC
- Communications are implemented through LLAPI using SCIF

Compiler pragma offload based
- API is using Phi-specific offload directives
- Enhancements for CPU-MIC communications

ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Scheduling strategies

No need to explicitly code data dependencies and data transfers. This is hidden in the runtime system.

High-productivity with Dynamic Runtime Systems

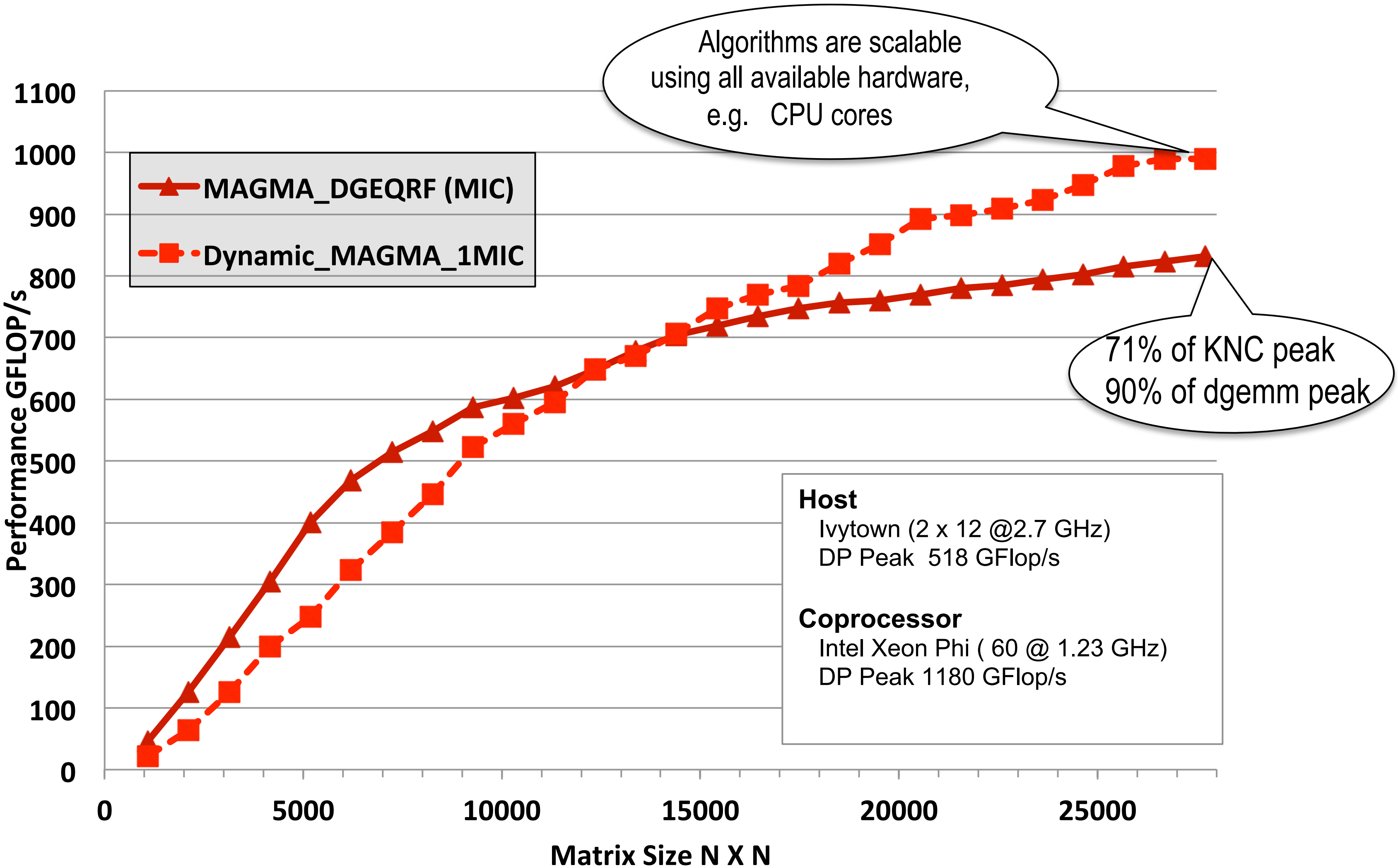
From sequential code → Parallel execution

```
for (k = 0; k < min(MT, NT); k++){
    zgeqrt(A[k;k], ...);
    for (n = k+1; n < NT; n++)
        zunmqr(A[k;k], A[k;n], ...);
    for (m = k+1; m < MT; m++){
        ztsqrt(A[k;k],,A[m;k], ...);
        for (n = k+1; n < NT; n++)
            ztsmqr(A[m;k], A[k;n], A[m;n], ...);
    }
}
```

```
for (k = 0; k < min(MT, NT); k++){
    Insert_Task(&zgeqrt, k , k, ...);
    for (n = k+1; n < NT; n++)
     Insert_Task(&zunmqr, k, n, ...);
    for (m = k+1; m < MT; m++){
     Insert_Task(&ztsqrt, m, k, ...);
     for (n = k+1; n < NT; n++)
         Insert_Task(&ztsmqr, m, n, k, ...);
    }
}
```


ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Performance on single MIC
QR AO with static and dynamic MAGMA

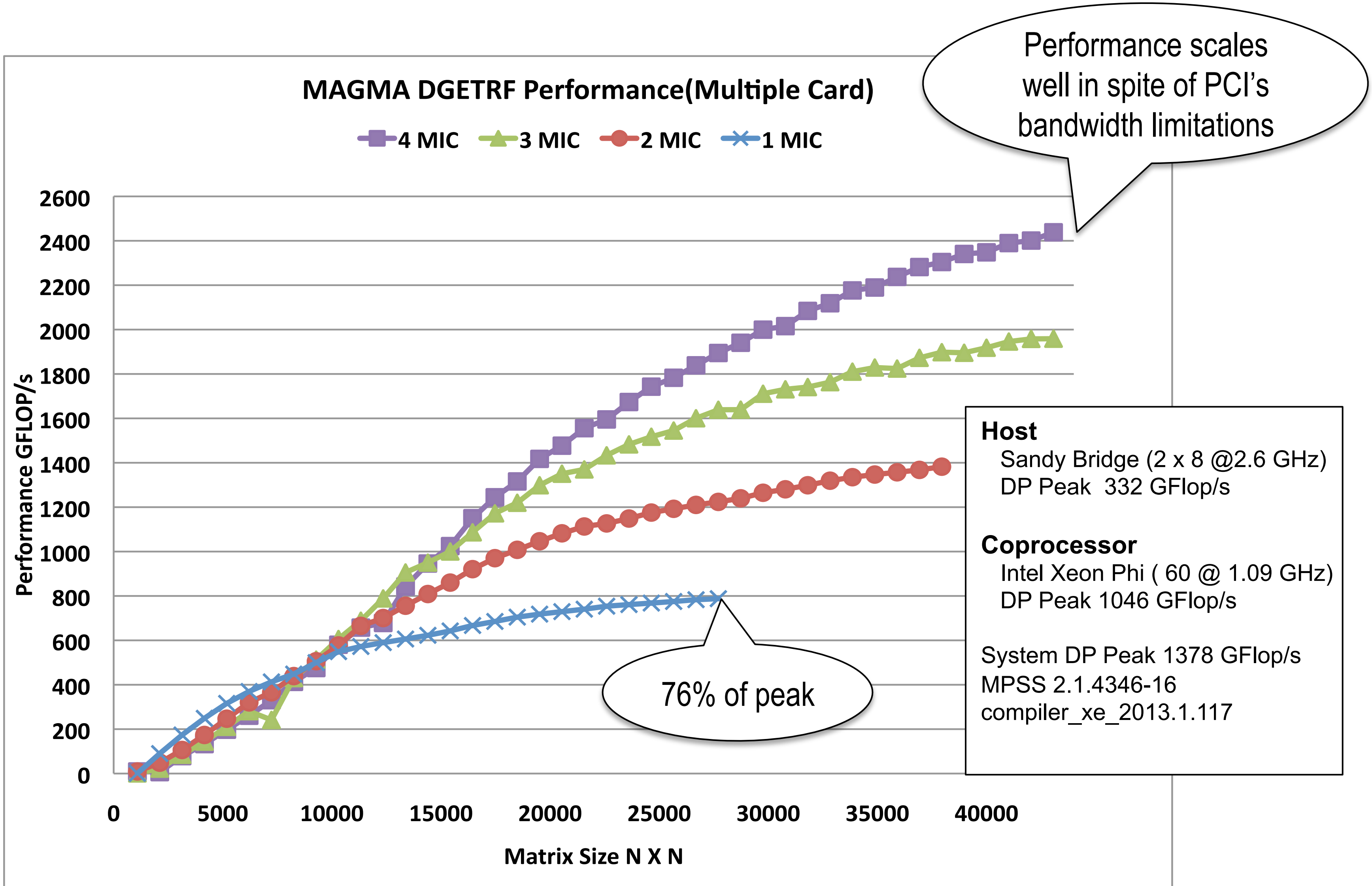


Host
Ivytown (2 x 12 @2.7 GHz)
DP Peak 518 GFlop/s

Coprocessor
Intel Xeon Phi (60 @ 1.23 GHz)
DP Peak 1180 GFlop/s

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Scalability on multiple MICs



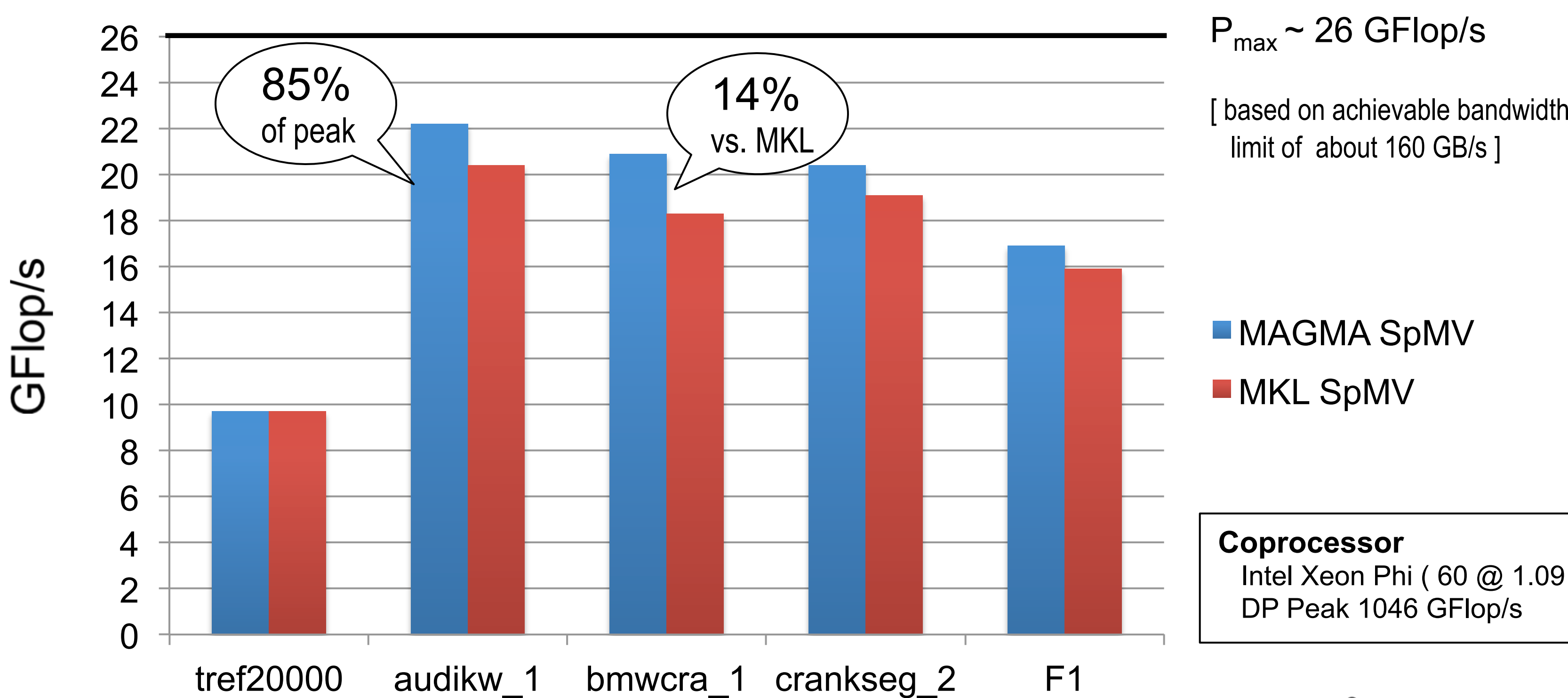
ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

MAGMA MIC Sparse

FEATURES AND SUPPORT in MAGMA MIC 1.4

ROUTINES	CG, GMRES, BiCGSTAB, Iterative Refinement
PRECONDITIONERS	Jacobi, user defined
KERNELS	SpMV, SpMM
DATA FORMATS	CSR, CPU converters from/to ELL and SELL-P
BENCHMARKS	CG, GMRES, BiCGSTAN, SpMV, SpMM, BLAS

PERFORMANCE (in double precision)



Coprocessor
Intel Xeon Phi (60 @ 1.09 GHz)
DP Peak 1046 GFlop/s

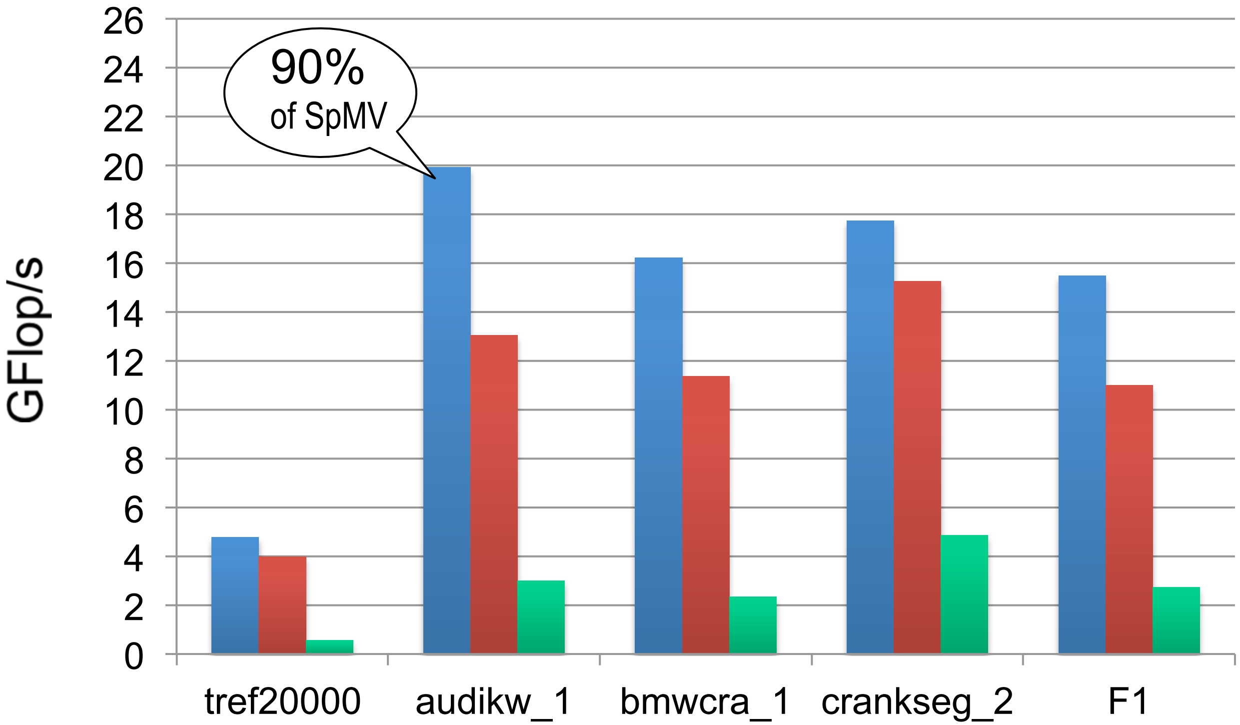
ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

MAGMA MIC Sparse

FEATURES AND SUPPORT in MAGMA MIC 1.4

ROUTINES	CG, GMRES, BiCGSTAB, Iterative Refinement
PRECONDITIONERS	Jacobi, user defined
KERNELS	SpMV, SpMM
DATA FORMATS	CSR, CPU converters from/to ELL and SELL-P
BENCHMARKS	CG, GMRES, BiCGSTAN, SpMV, SpMM, BLAS

PERFORMANCE (in double precision)



	Flops / iter.
CG	14n + SpMV
BiCGSTAB	20n + 2 SpMV
GMRES(30)	4(i+1) + SpMV

Coprocessor
Intel Xeon Phi (60 @ 1.09 GHz)
DP Peak 1046 GFlop/s

ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

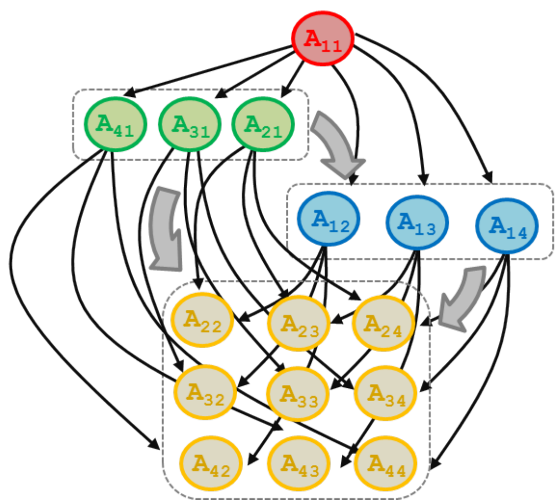
Future directions

- **Algorithms to increase the computational intensity of LA**
  - **“Communication-avoiding” type of algorithms**
  - **Mixed-precision LA**
  - **Applications/solvers that organize the computation to be on small dense matrices, usually data-independent, and many, that can be “batched”**
    - **Tiled linear algebra algorithms**
    - **Multifrontal methods**
    - **Preconditioners (using DLA) in sparse iterative solvers**
    - **Tensor contractions (in high-order FEM, etc.)**

Sparse / Dense Matrix System

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}$$

DAG-based factorization



To capture main LA patterns needed in a numerical library for Batched LA

- **LU, QR,** or **Cholesky** on small diagonal matrices
- **TRSMs**, **QRs**, or **LUs**
- **TRSMs**, **TRMMs**
- **Updates (Schur complement) GEMMs**, **SYRKs**, **TRMMs**

And many other BLAS/LAPACK, e.g., for application specific solvers, preconditioners, and matrices

Future directions

- Batched LA to provide support for various applications
- Communication-avoiding algorithms in dense and sparse LA applications
- Mixed-precision methods
- Benchmarks

Collaborators and Support

MAGMA team

http://icl.cs.utk.edu/magma

PLASMA team

http://icl.cs.utk.edu/plasma

Intel MKL team

Collaborating partners

University of Tennessee, Knoxville
University of California, Berkeley
University of Colorado, Denver
INRIA, France (StarPU team)
KAUST, Saudi Arabia